
# View Source: Design Patterns in the Wild

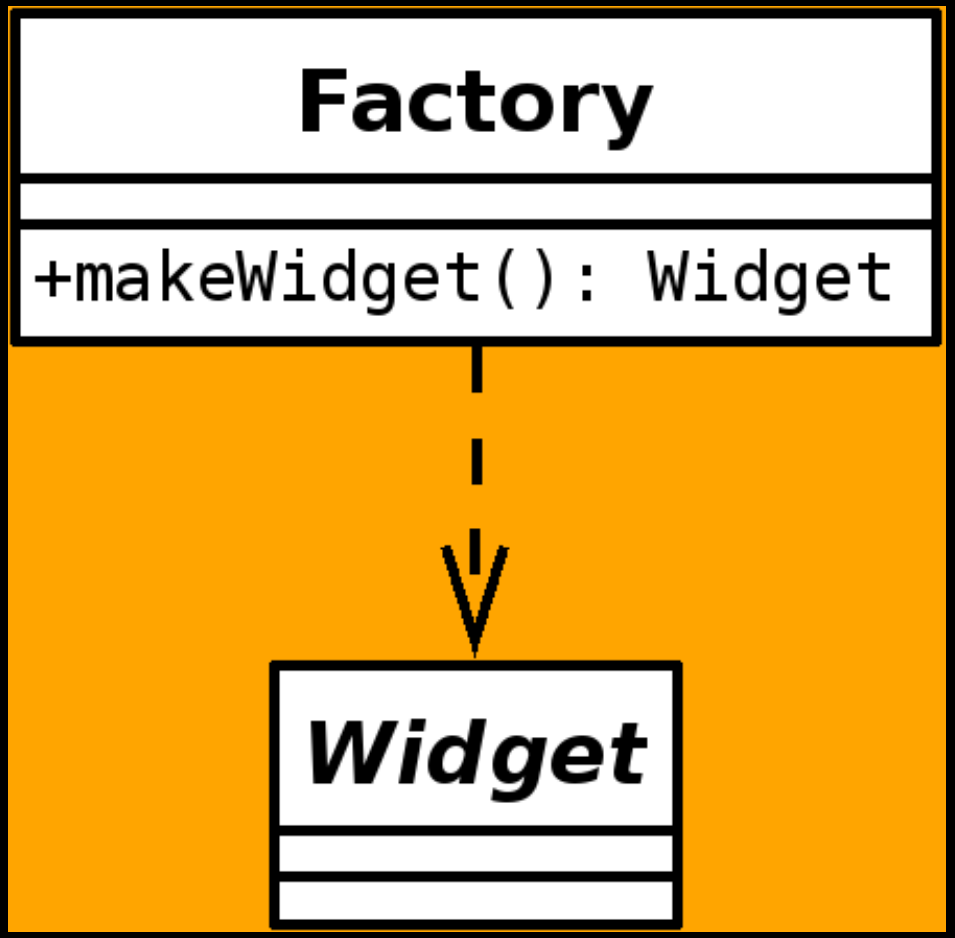
## Patterns in Log4J

Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

# Patterns for Today

- Factory
- Template Method
- Observer
- Mediator
- Facade

# Factory

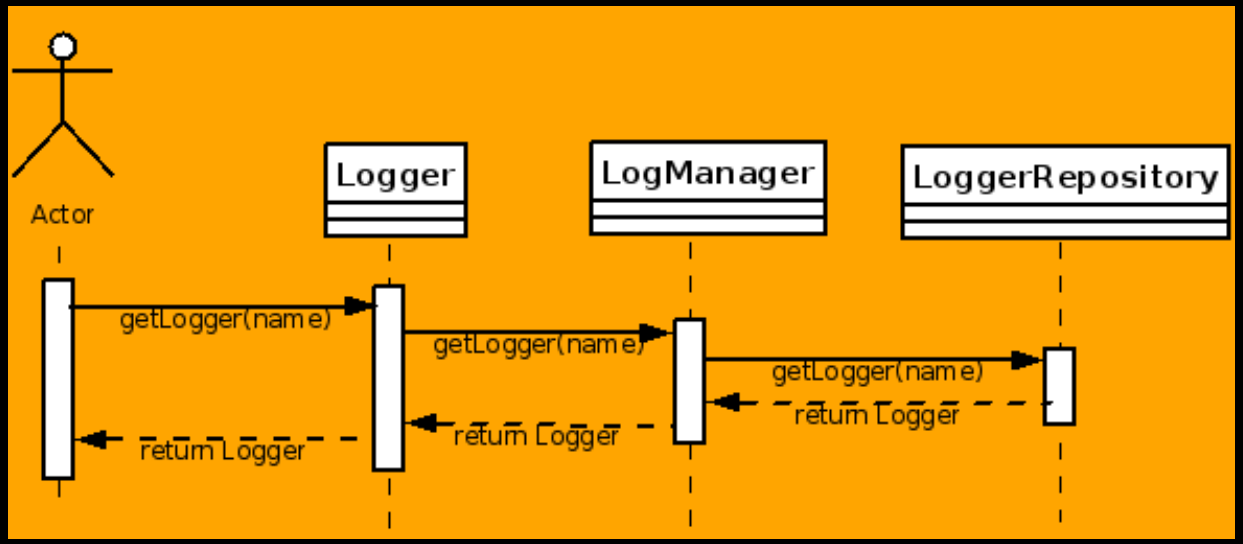


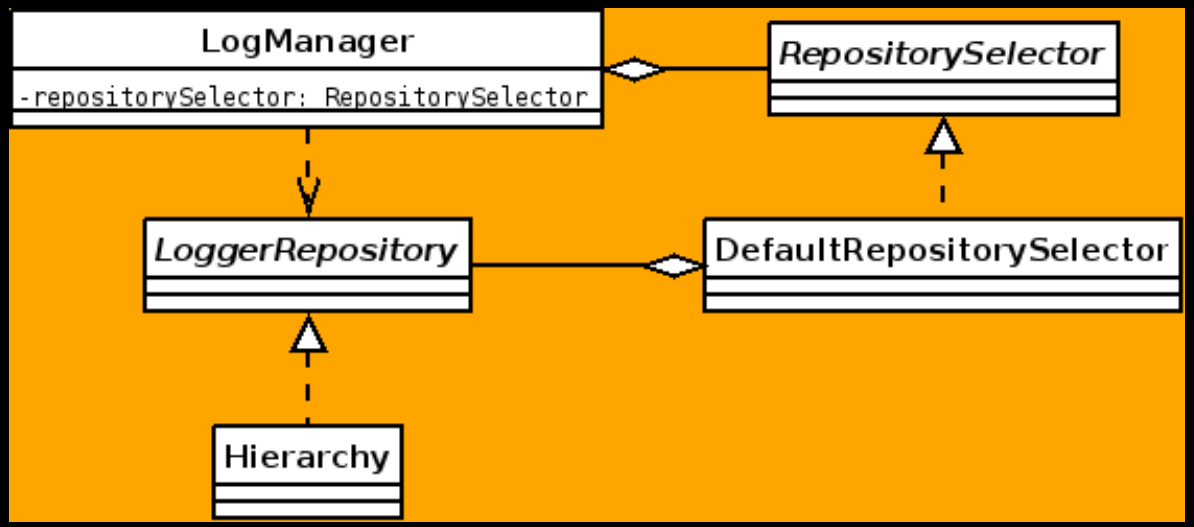
# new() vs. Factory



Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

# Logger Factory



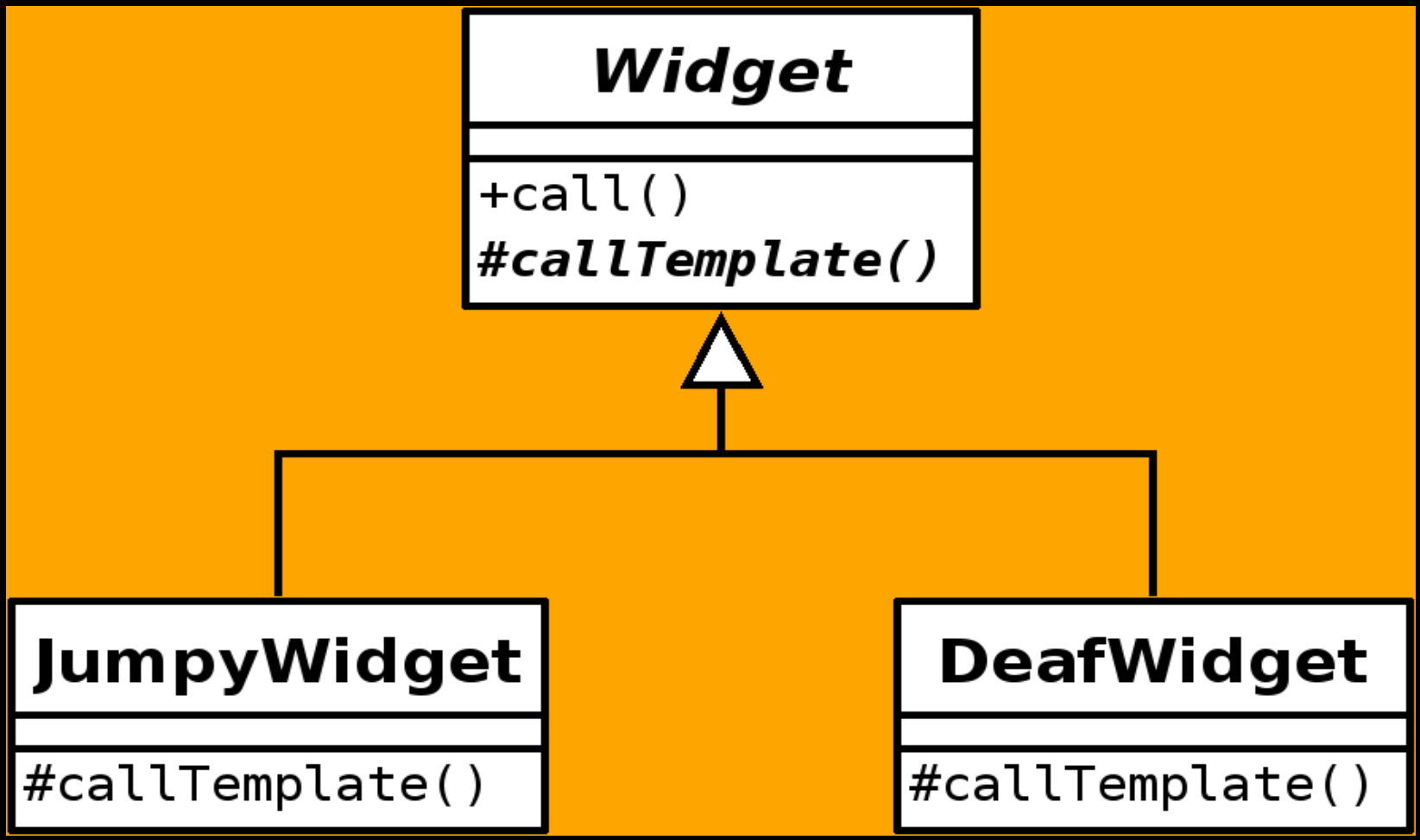




Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

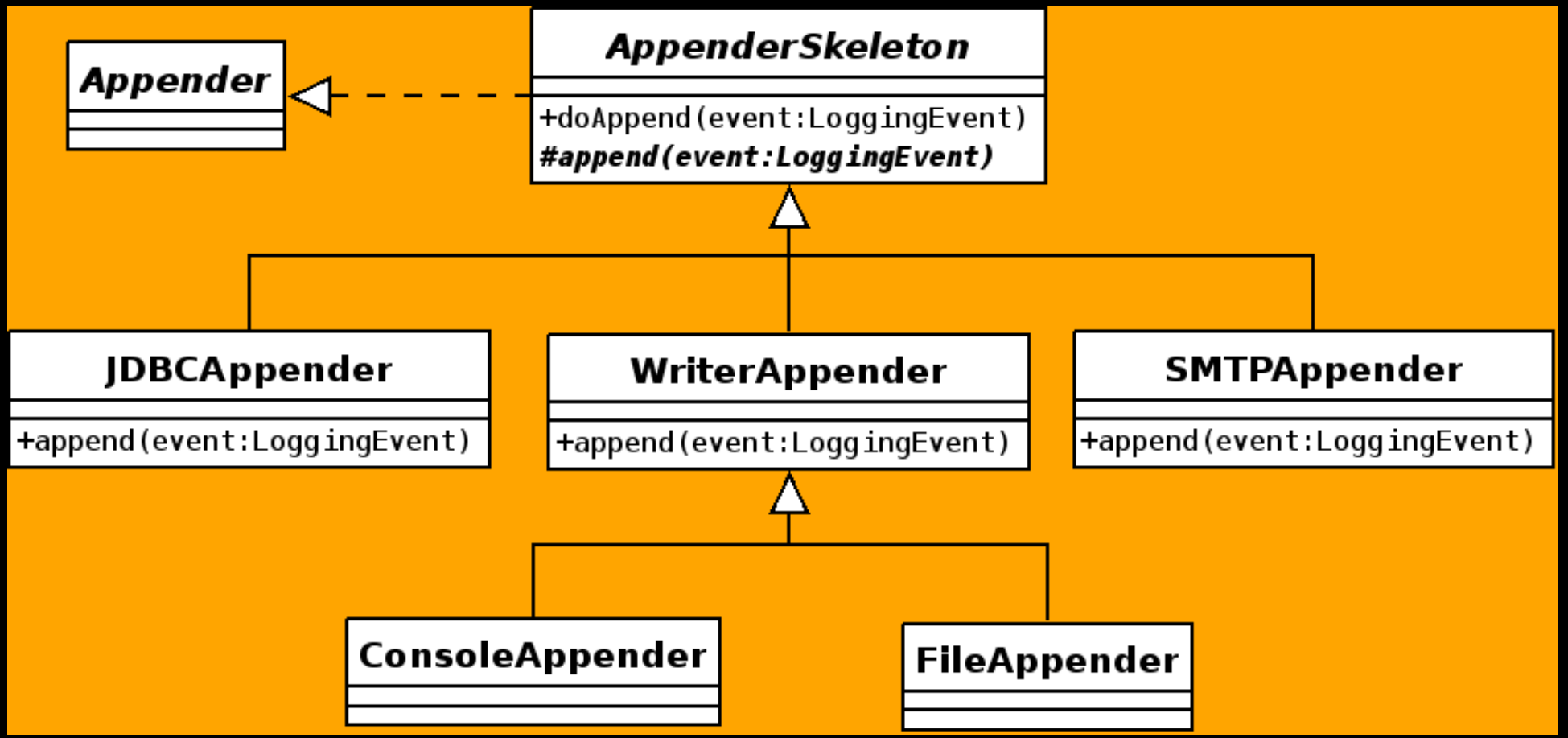
# Using Factory

- Functional
  - Construct objects from different classes depending on parameters
  - Return object instances from a cache
- Organizational
  - Break out complex constructor logic to a separate factory class

# Template Method



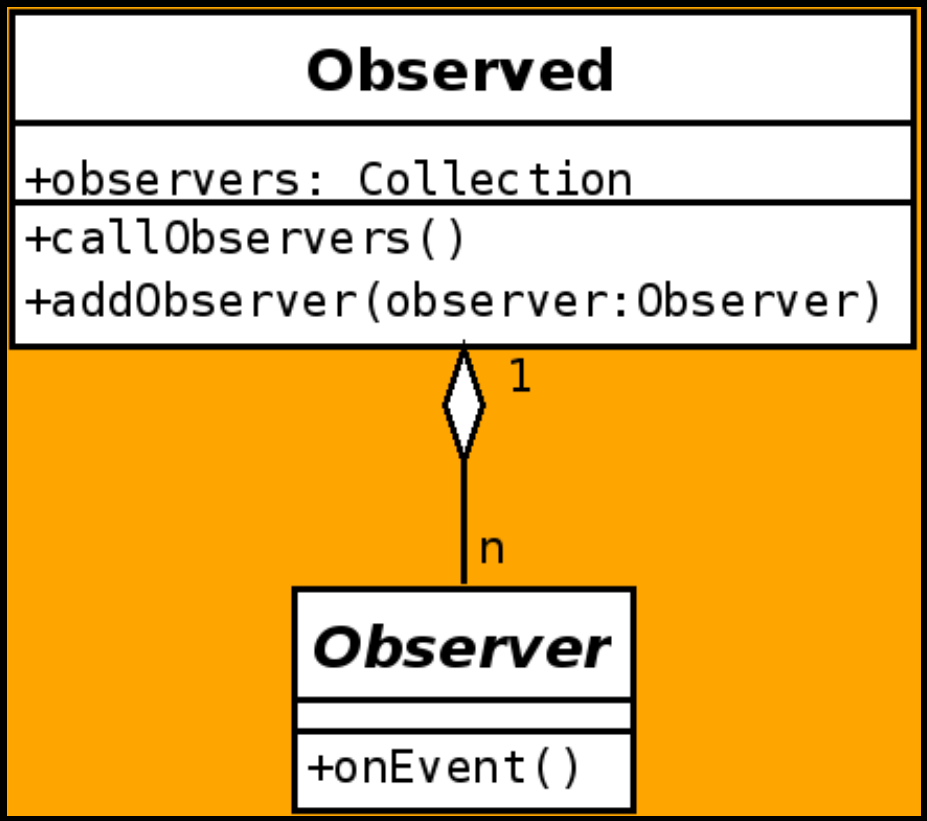
# Appender Template Method



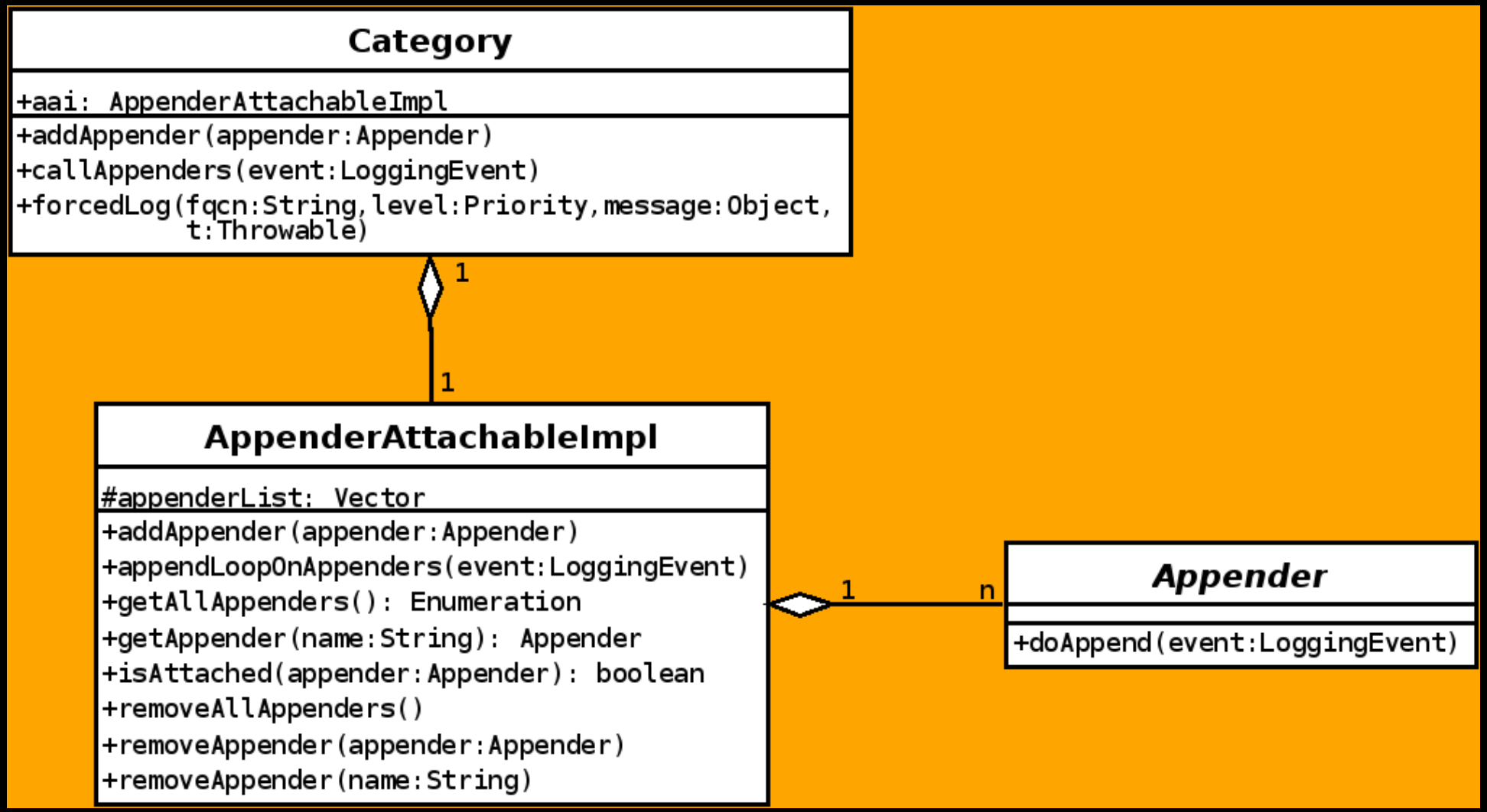
# Using Template Method

- Wrap common logic around specialized logic
- Concrete subclasses provide specialization for template method
- Conceptually similar to Aspect Oriented Programming (AOP)

# Observer


Observed
+observers: Collection
+callObservers()
+addObserver(observer:Observer)
1
n
Observer
+onEvent()

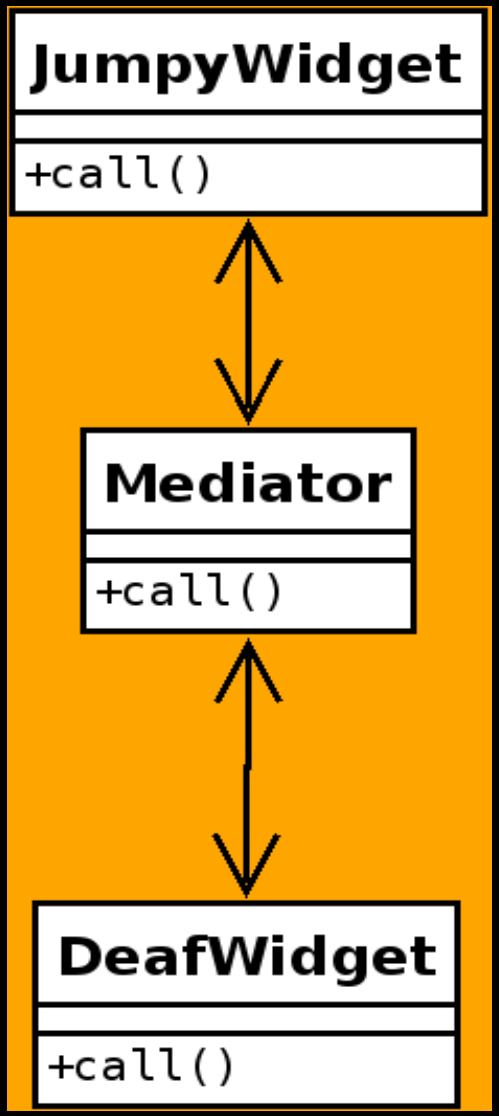
# Appenders as Observers



# Using Observer

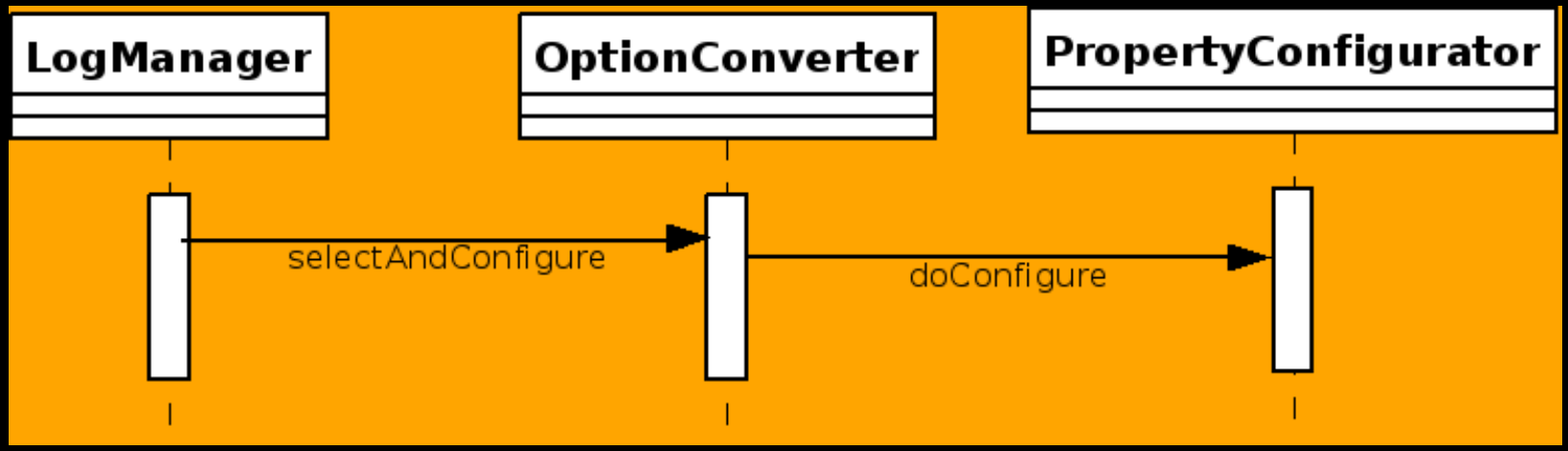
- When arbitrary numbers of objects need to be invoked on events in calling object
- Decouple observers from calling object

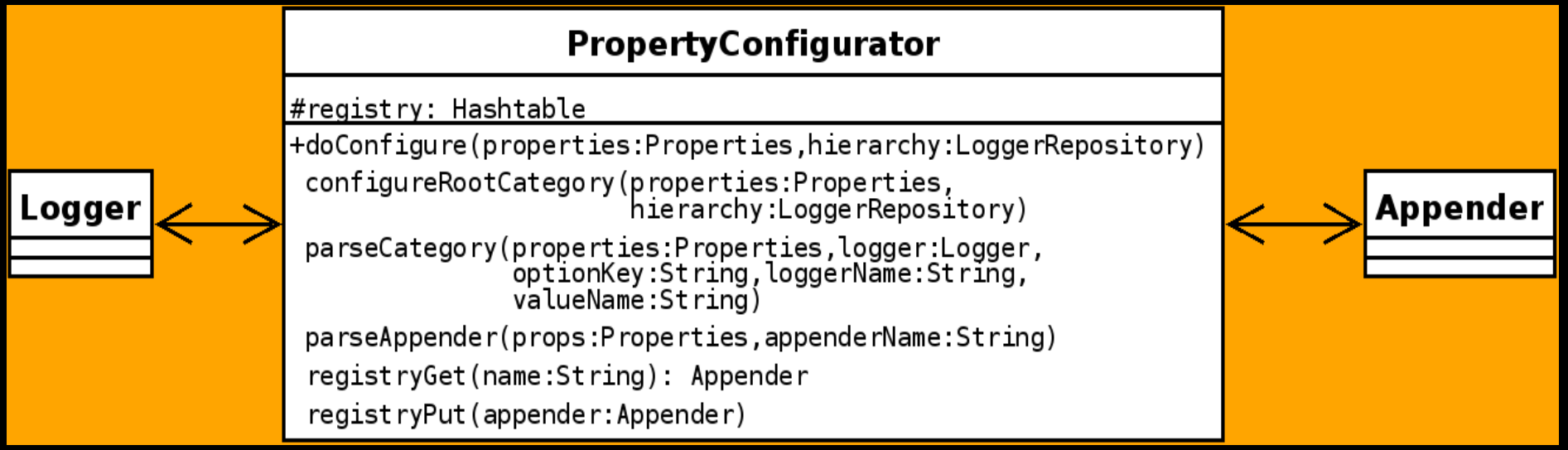
# Mediator




Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

# Mediating Loggers and Appenders



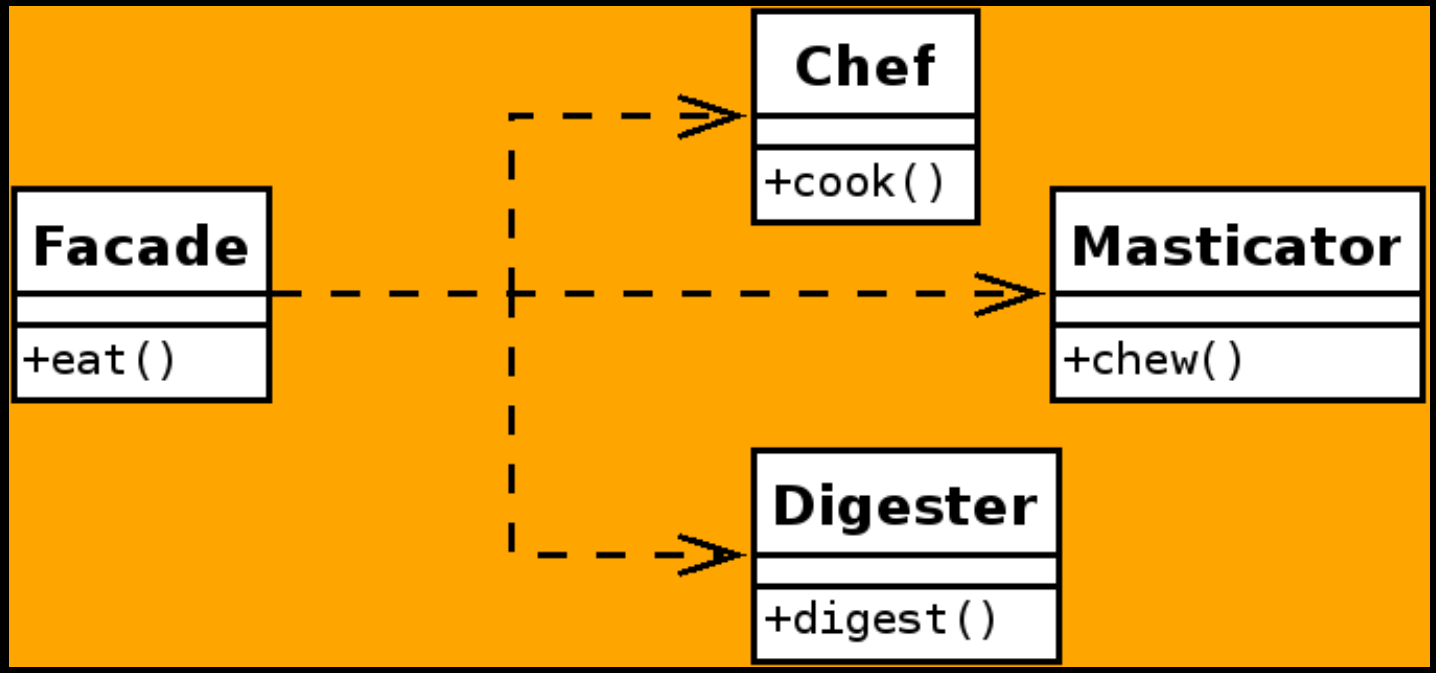




Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

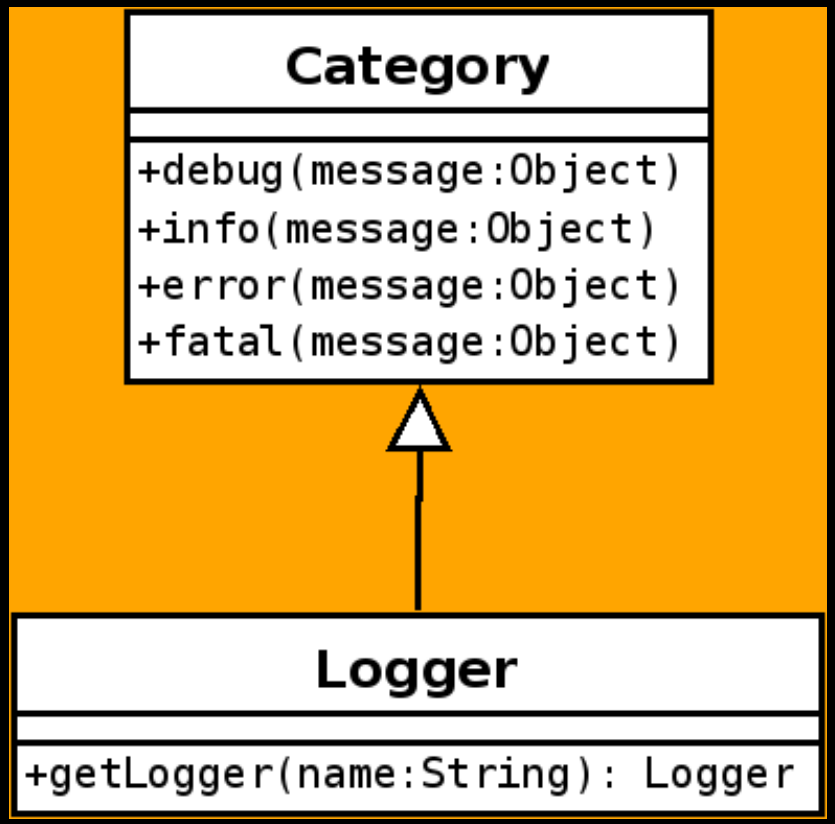
# Using Mediator

- Keep related classes from referring to one another directly
- Extract logic relating different classes to Mediator
- Warning: potential God class!

# Facade



# Logger as Facade



# Using Facade

- Simple interface to complex functionality
- Encapsulates a subsystem of classes
- Single class, or small group of classes